Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Sat, 05 Feb 2011 18:55:23 GMT
View Forum Message <> Reply to Message

Suka,

This patch - and the corresponding kernel patch - are wrong
(I should have noticed it in the review!). It turns out that
ghost (and dead) tasks are _not_ auto-reaped anymore.

There are only two way for tasks to be auto-reaped: one is if
their parent explicitly says so in its sighand information (but
then it applies to all children). The other way is if they have
->exit_signal==-1. From userspace this happens only when cloning
with CLONE_THREAD. Using 0xFF for the @flags argument to clone()
syscall instead results in ->exit_signal = 255 ...

The original motivation for this patch was:

> The downside of marking the task detached in do_ghost_task() is that
> with current/older kernels container-init does not wait for detached
> tasks. See:
>
> http://lkml.org/lkml/2010/6/16/272
> http://lkml.org/lkml/2010/7/12/213
>
> This can lead to a kernel crash if the container-init pre-deceases a
> ghost task.

Is this still a problem in 2.6.37 ?

Oren.

On 01/10/2011 08:51 PM, Oren Laadan wrote:
>
> Applied to user-cr.
>
> Thanks,
>
> Oren.
>
> On 12/10/2010 10:35 PM, Sukadev Bhattiprolu wrote:
>>
>> From: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
>> Date: Fri, 10 Dec 2010 19:23:58 -0800
>> Subject: [PATCH 1/1] Ghost tasks must be detached
>>

>> Ghost processes are created only to help restore orphaned sessions/pgrps.
>> As such once the session/pgrp is created the ghost must not send another
>> SIGCHLD to the parent but exit silently. So create such tasks as
>> "detached".
>>
>> See also:
>>
>> https://lists.linux-foundation.org/pipermail/containers/2010 -December/026076.html
>>
>> Signed-off-by: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
>> ---
>> restart.c | 7 +++++++
>> 1 files changed, 7 insertions(+), 0 deletions(-)
>>
>> diff --git a/restart.c b/restart.c
>> index 9fb5e9f..d7ba26b 100644
>> --- a/restart.c
>> +++ b/restart.c
>> @@ -1744,6 +1744,13 @@ static pid_t ckpt_fork_child(struct ckpt_ctx *ctx, struct task *child)
>>   flags |= CLONE_THREAD | CLONE_SIGHAND | CLONE_VM;
>>   else if (child->flags & TASK_SIBLING)
>>   flags |= CLONE_PARENT;
>> + else if (child->flags & (TASK_GHOST|TASK_DEAD)) {
>> + /*
>> +   * Ghosts must vanish silently (without signalling parent)
>> +   * when they are done.
>> +   */
>> + flags = 0xFF;
>> + }
>>
>>   memset(&clone_args, 0, sizeof(clone_args));
>>   clone_args.nr_pids = 1;
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containe rs
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Sat, 05 Feb 2011 21:40:32 GMT
View Forum Message <> Reply to Message

Oren Laadan [orenl@cs.columbia.edu] wrote:

| Suka,
|
| This patch - and the corresponding kernel patch - are wrong

Ah, I see that now.

But am not sure about the kernel part though. We were getting a crash
reliably (with older kernels) because of the ->exit_signal = -1 in
do_ghost_task().

One fix I was watching for was Eric Biederman's

 http://lkml.org/lkml/2010/7/12/213

which AFAICT has not been merged yet.

Was there another change to 2.6.37 that would prevent the crash ?

| (I should have noticed it in the review!). It turns out that
| ghost (and dead) tasks are _not_ auto-reaped anymore.
|
| There are only two way for tasks to be auto-reaped: one is if
| their parent explicitly says so in its sighand information (but
| then it applies to all children). The other way is if they have
| ->exit_signal==-1. From userspace this happens only when cloning
| with CLONE_THREAD. Using 0xFF for the @flags argument to clone()
| syscall instead results in ->exit_signal = 255 ...
|
| The original motivation for this patch was:
|
| > The downside of marking the task detached in do_ghost_task() is that
| > with current/older kernels container-init does not wait for detached
| > tasks. See:
| >
| > http://lkml.org/lkml/2010/6/16/272
| > http://lkml.org/lkml/2010/7/12/213
| >
| > This can lead to a kernel crash if the container-init pre-deceases a
| > ghost task.
|
| Is this still a problem in 2.6.37 ?

Well, some inadvertent userspace changes seemed to cause the crash (or
an application hang on some machines) during restart. By making those changes,
I seem to be getting an application hang 5 out of 6 times even with 2.6.37,
but did not get a crash. I will investigate this new hang next week.

|

| Oren.
|
|
| On 01/10/2011 08:51 PM, Oren Laadan wrote:
| >
| > Applied to user-cr.
| >
| > Thanks,
| >
| > Oren.
| >
| > On 12/10/2010 10:35 PM, Sukadev Bhattiprolu wrote:
| >>
| >> From: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
| >> Date: Fri, 10 Dec 2010 19:23:58 -0800
| >> Subject: [PATCH 1/1] Ghost tasks must be detached
| >>
| >> Ghost processes are created only to help restore orphaned sessions/pgrps.
| >> As such once the session/pgrp is created the ghost must not send another
| >> SIGCHLD to the parent but exit silently. So create such tasks as
| >> "detached".
| >>
| >> See also:
| >>
| >> https://lists.linux-foundation.org/pipermail/containers/2010 -December/026076.html
| >>
| >> Signed-off-by: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
| >> ---
| >> restart.c | 7 +++++++
| >> 1 files changed, 7 insertions(+), 0 deletions(-)
| >>
| >> diff --git a/restart.c b/restart.c
| >> index 9fb5e9f..d7ba26b 100644
| >> --- a/restart.c
| >> +++ b/restart.c
| >> @@ -1744,6 +1744,13 @@ static pid_t ckpt_fork_child(struct ckpt_ctx *ctx, struct task *child)
| >>   flags |= CLONE_THREAD | CLONE_SIGHAND | CLONE_VM;
| >>   else if (child->flags & TASK_SIBLING)
| >>   flags |= CLONE_PARENT;
| >> + else if (child->flags & (TASK_GHOST|TASK_DEAD)) {
| >> + /*
| >> +   * Ghosts must vanish silently (without signalling parent)
| >> +   * when they are done.
| >> +   */
| >> + flags = 0xFF;
| >> + }
| >>
| >>   memset(&clone_args, 0, sizeof(clone_args));

| >> clone_args.nr_pids = 1;
| > _____
| > Containers mailing list
| > Containers@lists.linux-foundation.org
| > https://lists.linux-foundation.org/mailman/listinfo/containe rs
| >

_____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Sat, 05 Feb 2011 22:02:51 GMT
View Forum Message <> Reply to Message

On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> | Suka,
> |
> | This patch - and the corresponding kernel patch - are wrong
>
> Ah, I see that now.
>
> But am not sure about the kernel part though. We were getting a crash
> reliably (with older kernels) because of the ->exit_signal = -1 in
> do_ghost_task().

Are we still getting it with 2.6.37 ?
>
> One fix I was watching for was Eric Biederman's
>
> http://lkml.org/lkml/2010/7/12/213
>
> which AFAICT has not been merged yet.

If we need it and it isn't in mainline (any reason why ?) then
we can just add it to our linux-cr tree, as a preparatory patch.

>
> Was there another change to 2.6.37 that would prevent the crash ?

I don't know whether *that* crash still happens in 2.6.37 -
because I still didn't test it with that kernel line back.
(Actually, I never experienced that crash here even with
earlier kernels).

>

> | (I should have noticed it in the review!). It turns out that
> | ghost (and dead) tasks are _not_ auto-reaped anymore.
> |
> | There are only two way for tasks to be auto-reaped: one is if
> | their parent explicitly says so in its sighand information (but
> | then it applies to all children). The other way is if they have
> | ->exit_signal==-1. From userspace this happens only when cloning
> | with CLONE_THREAD. Using 0xFF for the @flags argument to clone()
> | syscall instead results in ->exit_signal = 255 ...
> |
> | The original motivation for this patch was:
> |
> | > The downside of marking the task detached in do_ghost_task() is that
> | > with current/older kernels container-init does not wait for detached
> | > tasks. See:
> | >
> | > http://lkml.org/lkml/2010/6/16/272
> | > http://lkml.org/lkml/2010/7/12/213
> | >
> | > This can lead to a kernel crash if the container-init pre-deceases a
> | > ghost task.
> |
> | Is this still a problem in 2.6.37 ?
>
> Well, some inadvertent userspace changes seemed to cause the crash (or
> an application hang on some machines) during restart. By making those changes,
> I seem to be getting an application hang 5 out of 6 times even with 2.6.37,
> but did not get a crash. I will investigate this new hang next week.

I'm currently chasing down a bug that causes restart to hang when
there are ghost/dead tasks. It may be the same one you are seeing.
So far I'm convinced it's userspace - working on it. Will post
patches once solved.

Thanks,

Oren.

>
> |
> | Oren.
> |
> |
> | On 01/10/2011 08:51 PM, Oren Laadan wrote:
> | >
> | > Applied to user-cr.
> | >

> | > Thanks,
> | >
> | > Oren.
> | >
> | > On 12/10/2010 10:35 PM, Sukadev Bhattiprolu wrote:
> | >>
> | >> From: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
> | >> Date: Fri, 10 Dec 2010 19:23:58 -0800
> | >> Subject: [PATCH 1/1] Ghost tasks must be detached
> | >>
> | >> Ghost processes are created only to help restore orphaned sessions/pgrps.
> | >> As such once the session/pgrp is created the ghost must not send another
> | >> SIGCHLD to the parent but exit silently. So create such tasks as
> | >> "detached".
> | >>
> | >> See also:
> | >>
> | >> https://lists.linux-foundation.org/pipermail/containers/2010 -December/026076.html
> | >>
> | >> Signed-off-by: Sukadev Bhattiprolu <sukadev@linux.vnet.ibm.com>
> | >> ---
> | >> restart.c | 7 +++++++
> | >> 1 files changed, 7 insertions(+), 0 deletions(-)
> | >>
> | >> diff --git a/restart.c b/restart.c
> | >> index 9fb5e9f..d7ba26b 100644
> | >> --- a/restart.c
> | >> +++ b/restart.c
> | >> @@ -1744,6 +1744,13 @@ static pid_t ckpt_fork_child(struct ckpt_ctx *ctx, struct task
*child)
> | >>   flags |= CLONE_THREAD | CLONE_SIGHAND | CLONE_VM;
> | >>   else if (child->flags & TASK_SIBLING)
> | >>   flags |= CLONE_PARENT;
> | >> + else if (child->flags & (TASK_GHOST|TASK_DEAD)) {
> | >> + /*
> | >> +   * Ghosts must vanish silently (without signalling parent)
> | >> +   * when they are done.
> | >> +   */
> | >> + flags = 0xFF;
> | >> + }
> | >>
> | >>   memset(&clone_args, 0, sizeof(clone_args));
> | >>   clone_args.nr_pids = 1;
> | > _____
> | > Containers mailing list
> | > Containers@lists.linux-foundation.org
> | > https://lists.linux-foundation.org/mailman/listinfo/containe rs
> | >

>

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Sat, 05 Feb 2011 22:33:15 GMT
View Forum Message <> Reply to Message

On 02/05/2011 05:02 PM, Oren Laadan wrote:
>
>
> On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
>> Oren Laadan [orenl@cs.columbia.edu] wrote:
>> | Suka,
>> |
>> | This patch - and the corresponding kernel patch - are wrong
>>
>> Ah, I see that now.
>>
>> But am not sure about the kernel part though. We were getting a crash
>> reliably (with older kernels) because of the ->exit_signal = -1 in
>> do_ghost_task().
>
> Are we still getting it with 2.6.37 ?
>>
>> One fix I was watching for was Eric Biederman's
>>
>> http://lkml.org/lkml/2010/7/12/213
>>
>> which AFAICT has not been merged yet.
>
> If we need it and it isn't in mainline (any reason why ?) then
> we can just add it to our linux-cr tree, as a preparatory patch.
>
>>
>> Was there another change to 2.6.37 that would prevent the crash ?
>
> I don't know whether *that* crash still happens in 2.6.37 -
> because I still didn't test it with that kernel line back.
> (Actually, I never experienced that crash here even with
> earlier kernels).
>
>>
>> | (I should have noticed it in the review!). It turns out that
>> | ghost (and dead) tasks are _not_ auto-reaped anymore.

>> |
>> | There are only two way for tasks to be auto-reaped: one is if
>> | their parent explicitly says so in its sighand information (but
>> | then it applies to all children). The other way is if they have
>> | ->exit_signal==-1. From userspace this happens only when cloning
>> | with CLONE_THREAD. Using 0xFF for the @flags argument to clone()
>> | syscall instead results in ->exit_signal = 255 ...
>> |
>> | The original motivation for this patch was:
>> |
>> | > The downside of marking the task detached in do_ghost_task() is that
>> | > with current/older kernels container-init does not wait for detached
>> | > tasks. See:
>> | >
>> | > http://lkml.org/lkml/2010/6/16/272
>> | > http://lkml.org/lkml/2010/7/12/213
>> | >
>> | > This can lead to a kernel crash if the container-init pre-deceases a
>> | > ghost task.
>> |
>> | Is this still a problem in 2.6.37 ?
>>
>> Well, some inadvertent userspace changes seemed to cause the crash (or
>> an application hang on some machines) during restart. By making those changes,
>> I seem to be getting an application hang 5 out of 6 times even with 2.6.37,
>> but did not get a crash. I will investigate this new hang next week.
>
> I'm currently chasing down a bug that causes restart to hang when
> there are ghost/dead tasks. It may be the same one you are seeing.
> So far I'm convinced it's userspace - working on it. Will post
> patches once solved.

Actually - scratch that. The bug was related to the new pids-as-objs
patches to user-cr.

Now, given the complexity of that patchset, I don't want to spend
time on picking fixes that require backporting to the current user-cr.
So let's try to focus testing and debugging efforts on kernel/user
that includes those patches ?

Thanks,

Oren.

_____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Wed, 09 Feb 2011 02:09:43 GMT
View Forum Message <> Reply to Message

Oren Laadan [orenl@cs.columbia.edu] wrote:
|
|
| On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
| > Oren Laadan [orenl@cs.columbia.edu] wrote:
| > | Suka,
| > |
| > | This patch - and the corresponding kernel patch - are wrong
| >
| > Ah, I see that now.
| >
| > But am not sure about the kernel part though. We were getting a crash
| > reliably (with older kernels) because of the ->exit_signal = -1 in
| > do_ghost_task().
|
| Are we still getting it with 2.6.37 ?

I am not currently getting the crash on 2.6.37 - I thought it was due to
the following commit which removed the check for task_detached() in
do_wait_thread().

 commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
 Author: Oleg Nesterov <oleg@redhat.com>
 Date: Thu Dec 17 15:27:15 2009 -0800

But if that is true, I need to investigate why Louis Rilling was getting
the crash in Jun 2010 - which he tried to fix here:

 http://lkml.org/lkml/2010/6/16/295

Even if we are not currently not getting the crash, I think user-space
actions can result in the container-init being unable to forcibly kill
all its children and exit.

Eg: if ghost tasks are pushed into a child pid namespace (by intentionally
setting ->piddepth in usercr/restart.c), we can have a situation where the
ghost task exits silently, the parent (i.e container-init can be left hanging).

It can be argued that the incorrect changes in usercr code result in the
application hang.

But pid namespace is supposed to guarantee that if a container-init is
terminated, it will take the pid namespace down. But some userspace
actions can result in kill -9 of container-init leaving the container-init
hung forever.

| >
| > One fix I was watching for was Eric Biederman's
| >
| > http://lkml.org/lkml/2010/7/12/213
| >
| > which AFAICT has not been merged yet.
|
| If we need it and it isn't in mainline (any reason why ?) then
| we can just add it to our linux-cr tree, as a preparatory patch.
|
| >
| > Was there another change to 2.6.37 that would prevent the crash ?
|
| I don't know whether *that* crash still happens in 2.6.37 -
| because I still didn't test it with that kernel line back.
| (Actually, I never experienced that crash here even with
| earlier kernels).

Yes, it needed some "accidental" usercr change to expose the crash :-)

(I will try to send a patch to existing usercr and a test case to repro
this problem)

_____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Wed, 09 Feb 2011 03:35:20 GMT
View Forum Message <> Reply to Message

On 02/08/2011 09:09 PM, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> |
> |
> | On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
> | > Oren Laadan [orenl@cs.columbia.edu] wrote:
> | > | Suka,
> | > |
> | > | This patch - and the corresponding kernel patch - are wrong
> | >
> | > Ah, I see that now.
> | >
> | > But am not sure about the kernel part though. We were getting a crash
> | > reliably (with older kernels) because of the ->exit_signal = -1 in

> | > do_ghost_task().
> |
> | Are we still getting it with 2.6.37 ?
>
> I am not currently getting the crash on 2.6.37 - I thought it was due to
> the following commit which removed the check for task_detached() in
> do_wait_thread().
>
> commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
> Author: Oleg Nesterov <oleg@redhat.com>
> Date: Thu Dec 17 15:27:15 2009 -0800
>
> But if that is true, I need to investigate why Louis Rilling was getting
> the crash in Jun 2010 - which he tried to fix here:
>
> http://lkml.org/lkml/2010/6/16/295

I see. So basically there is a kerenl bug that can be potentially
exposed by the c/r code. Therefore, we need to fix the kernel bug...
(and until such a fix makes it to mainline, we'll add it as part of
the linux-cr patchset).

>
> Even if we are not currently not getting the crash, I think user-space
> actions can result in the container-init being unable to forcibly kill
> all its children and exit.
>
> Eg: if ghost tasks are pushed into a child pid namespace (by intentionally
> setting ->piddepth in usercr/restart.c), we can have a situation where the
> ghost task exits silently, the parent (i.e container-init can be left hanging).

I don't quite understand what you mean here. Basically, the ghost tasks
are only alive _during_ restart and are gone when the restart completes.
Therefore they cannot affect whether the init task of the new pidns will
hang or terminate -- that init task has no knowledge of ghost tasks.

Let's consider the two possible scenarios:
(1) container restart (that includes the container init)
(2) subtree restart in a new pidns (that does not include the init task,
and instead user-cr provides an init process to hold the new pidns alive).

Case 1: the (restarted) init task was part of the checkpoint. Typically
it would "wait()" in a loop for children until it gets ECHILD and then
exits (the container). Ghost tasks are not a factor here.

Case 2: the (injected) init task was not part of the checkpoint. It does
the same as the typical init in case 1: loop until wait() says no more
children, then exits. In this case, there will be at least one child of

that init task, because at least one task was restarted ... Typically,
when that child exits, our injected init task will exit. Again, ghost
tasks do no participate.

>
> It can be argued that the incorrect changes in usercr code result in the
> application hang.
>
> But pid namespace is supposed to guarantee that if a container-init is
> terminated, it will take the pid namespace down. But some userspace
> actions can result in kill -9 of container-init leaving the container-init
> hung forever.

So I guess I don't quite understand the concern. Can you describe a
concrete example ?

>
> | >
> | > One fix I was watching for was Eric Biederman's
> | >
> | > http://lkml.org/lkml/2010/7/12/213
> | >
> | > which AFAICT has not been merged yet.
> |
> | If we need it and it isn't in mainline (any reason why ?) then
> | we can just add it to our linux-cr tree, as a preparatory patch.
> |
> | >
> | > Was there another change to 2.6.37 that would prevent the crash ?
> |
> | I don't know whether *that* crash still happens in 2.6.37 -
> | because I still didn't test it with that kernel line back.
> | (Actually, I never experienced that crash here even with
> | earlier kernels).
>
> Yes, it needed some "accidental" usercr change to expose the crash :-)
>
> (I will try to send a patch to existing usercr and a test case to repro
> this problem)
>

Thanks,

Oren.

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Wed, 09 Feb 2011 12:01:00 GMT
View Forum Message <> Reply to Message

On 08/02/11 18:09 -0800, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> |
> |
> | On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
> | > Oren Laadan [orenl@cs.columbia.edu] wrote:
> | > | Suka,
> | > |
> | > | This patch - and the corresponding kernel patch - are wrong
> | >
> | > Ah, I see that now.
> | >
> | > But am not sure about the kernel part though. We were getting a crash
> | > reliably (with older kernels) because of the ->exit_signal = -1 in
> | > do_ghost_task().
> |
> | Are we still getting it with 2.6.37 ?
>
> I am not currently getting the crash on 2.6.37 - I thought it was due to
> the following commit which removed the check for task_detached() in
> do_wait_thread().
>
> commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
> Author: Oleg Nesterov <oleg@redhat.com>
> Date: Thu Dec 17 15:27:15 2009 -0800

I don't think that this introduced the bug. The bug triggers with EXIT_DEAD
tasks, for which wait() must ignore (see below). So, the bug looks still there
in 2.6.37.

>
> But if that is true, I need to investigate why Louis Rilling was getting
> the crash in Jun 2010 - which he tried to fix here:
>
> http://lkml.org/lkml/2010/6/16/295

I was getting the crash on Kerrighed, which heavily patches the 2.6.30 kernel.
I could reproduce it on vanilla Linux of the moment (2.6.35-rc3), but
only after introducing artificial delays in release_task().

IIRC, what triggers the crash is some exiting detached task in the

pid_namespace, which goes EXIT_DEAD, and as such cannot be reaped by
zap_pid_ns_processes()->sys_wait4(). So with some odd timing, the detached
task can call proc_flush_task() after container init does, which triggers the
proc_mnt crash.

```
Container init                      Some detached task in the ctnr
                                 exit_notify()
     ->exit_state = EXIT_DEAD
exit_notify()
 forget_original_parent()
  find_new_reaper()
   zap_pid_ns_processes()
    sys_wait4()
     /* cannot reap EXIT_DEAD tasks */
 /* reparents EXIT_DEAD tasks to global init */

Container reaper
release_task()
 proc_flush_task()
  pid_ns_release_proc()
                                 release_task()
                                  proc_flush_task()
                                   proc_flush_task_mnt()
                                    KABOOM
```

Thanks,

Louis

>
> Even if we are not currently not getting the crash, I think user-space
> actions can result in the container-init being unable to forcibly kill
> all its children and exit.
>
> Eg: if ghost tasks are pushed into a child pid namespace (by intentionally
> setting ->piddepth in usercr/restart.c), we can have a situation where the
> ghost task exits silently, the parent (i.e container-init can be left hanging).
>
> It can be argued that the incorrect changes in usercr code result in the
> application hang.
>
> But pid namespace is supposed to guarantee that if a container-init is
> terminated, it will take the pid namespace down. But some userspace
> actions can result in kill -9 of container-init leaving the container-init
> hung forever.
>
> | >
> | > One fix I was watching for was Eric Biederman's

> | >
> | > http://lkml.org/lkml/2010/7/12/213
> | >
> | > which AFAICT has not been merged yet.
> |
> | If we need it and it isn't in mainline (any reason why ?) then
> | we can just add it to our linux-cr tree, as a preparatory patch.
> |
> | >
> | > Was there another change to 2.6.37 that would prevent the crash ?
> |
> | I don't know whether *that* crash still happens in 2.6.37 -
> | because I still didn't test it with that kernel line back.
> | (Actually, I never experienced that crash here even with
> | earlier kernels).
>
> Yes, it needed some "accidental" usercr change to expose the crash :-)
>
> (I will try to send a patch to existing usercr and a test case to repro
> this problem)
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containe rs

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Wed, 09 Feb 2011 12:18:49 GMT
View Forum Message <> Reply to Message

On 02/09/2011 07:01 AM, Louis Rilling wrote:
> On 08/02/11 18:09 -0800, Sukadev Bhattiprolu wrote:
>> Oren Laadan [orenl@cs.columbia.edu] wrote:
>> |
>> |
>> | On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:

>> | > Oren Laadan [orenl@cs.columbia.edu] wrote:
>> | > | Suka,
>> | > |
>> | > | This patch - and the corresponding kernel patch - are wrong
>> | >
>> | > Ah, I see that now.
>> | >
>> | > But am not sure about the kernel part though. We were getting a crash
>> | > reliably (with older kernels) because of the ->exit_signal = -1 in
>> | > do_ghost_task().
>> |
>> | Are we still getting it with 2.6.37 ?
>>
>> I am not currently getting the crash on 2.6.37 - I thought it was due to
>> the following commit which removed the check for task_detached() in
>> do_wait_thread().
>>
>> commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
>> Author: Oleg Nesterov <oleg@redhat.com>
>> Date: Thu Dec 17 15:27:15 2009 -0800
>
> I don't think that this introduced the bug. The bug triggers with EXIT_DEAD
> tasks, for which wait() must ignore (see below). So, the bug looks still there
> in 2.6.37.
>
>>
>> But if that is true, I need to investigate why Louis Rilling was getting
>> the crash in Jun 2010 - which he tried to fix here:
>>
>> http://lkml.org/lkml/2010/6/16/295
>
> I was getting the crash on Kerrighed, which heavily patches the 2.6.30 kernel.
> I could reproduce it on vanilla Linux of the moment (2.6.35-rc3), but
> only after introducing artificial delays in release_task().
>
> IIRC, what triggers the crash is some exiting detached task in the
> pid_namespace, which goes EXIT_DEAD, and as such cannot be reaped by
> zap_pid_ns_processes()->sys_wait4(). So with some odd timing, the detached
> task can call proc_flush_task() after container init does, which triggers the
> proc_mnt crash.
>
> Container init   Some detached task in the ctnr
>   exit_notify()
>   ->exit_state = EXIT_DEAD
> exit_notify()
> forget_original_parent()
>   find_new_reaper()
>   zap_pid_ns_processes()

>   sys_wait4()
>   /* cannot reap EXIT_DEAD tasks */
> /* reparents EXIT_DEAD tasks to global init */
>
> Container reaper
> release_task()
> proc_flush_task()
>   pid_ns_release_proc()
>   release_task()
>   proc_flush_task()
>   proc_flush_task_mnt()
>   KABOOM

Louis, thanks for the explanation, and two follow-up questions:

1) Is there a patch circulating for this ? or even better, on the
way to mainline ?

2) Would it suffice if the c/r code ensures that the init never
exits before any EXIT_DEAD tasks ?

Thanks,

Oren.

_____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Wed, 09 Feb 2011 12:35:50 GMT
View Forum Message <> Reply to Message

On 09/02/11 7:18 -0500, Oren Laadan wrote:
>
>
> On 02/09/2011 07:01 AM, Louis Rilling wrote:
> > On 08/02/11 18:09 -0800, Sukadev Bhattiprolu wrote:
> >> Oren Laadan [orenl@cs.columbia.edu] wrote:
> >> |
> >> |
> >> | On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
> >> | > Oren Laadan [orenl@cs.columbia.edu] wrote:
> >> | > | Suka,
> >> | > |
> >> | > | This patch - and the corresponding kernel patch - are wrong
> >> | >

> >> | > Ah, I see that now.
> >> | >
> >> | > But am not sure about the kernel part though. We were getting a crash
> >> | > reliably (with older kernels) because of the ->exit_signal = -1 in
> >> | > do_ghost_task().
> >> |
> >> | Are we still getting it with 2.6.37 ?
> >>
> >> I am not currently getting the crash on 2.6.37 - I thought it was due to
> >> the following commit which removed the check for task_detached() in
> >> do_wait_thread().
> >>
> >> commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
> >> Author: Oleg Nesterov <oleg@redhat.com>
> >> Date: Thu Dec 17 15:27:15 2009 -0800
> >
> > I don't think that this introduced the bug. The bug triggers with EXIT_DEAD
> > tasks, for which wait() must ignore (see below). So, the bug looks still there
> > in 2.6.37.
> >
> >>
> >> But if that is true, I need to investigate why Louis Rilling was getting
> >> the crash in Jun 2010 - which he tried to fix here:
> >>
> >> http://lkml.org/lkml/2010/6/16/295
> >
> > I was getting the crash on Kerrighed, which heavily patches the 2.6.30 kernel.
> > I could reproduce it on vanilla Linux of the moment (2.6.35-rc3), but
> > only after introducing artificial delays in release_task().
> >
> > IIRC, what triggers the crash is some exiting detached task in the
> > pid_namespace, which goes EXIT_DEAD, and as such cannot be reaped by
> > zap_pid_ns_processes()->sys_wait4(). So with some odd timing, the detached
> > task can call proc_flush_task() after container init does, which triggers the
> > proc_mnt crash.
> >
> > Container init   Some detached task in the ctnr
> >   exit_notify()
> >   ->exit_state = EXIT_DEAD
> > exit_notify()
> > forget_original_parent()
> >   find_new_reaper()
> >   zap_pid_ns_processes()
> >   sys_wait4()
> >   /* cannot reap EXIT_DEAD tasks */
> > /* reparents EXIT_DEAD tasks to global init */
> >
> > Container reaper

> > release_task()
> > proc_flush_task()
> >   pid_ns_release_proc()
> >   release_task()
> >   proc_flush_task()
> >   proc_flush_task_mnt()
> >   KABOOM
>
> Louis, thanks for the explanation, and two follow-up questions:
>
> 1) Is there a patch circulating for this ? or even better, on the
> way to mainline ?

We finally agreed on a patch from Eric, but for some unknown reason, it has not been finalized(?) and routed to mainline yet.

https://lkml.org/lkml/2010/7/12/213

>
> 2) Would it suffice if the c/r code ensures that the init never
> exits before any EXIT_DEAD tasks ?

That's what Eric's patch does: make zap_pid_ns_processes() wait until all other tasks (EXIT_DEAD or whatever) have passed release_task()->__exit_signal()->__unhash_process().

Thanks,

Louis

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Wed, 09 Feb 2011 12:37:37 GMT
View Forum Message <> Reply to Message

Oops, failed to add Eric properly in Cc...

Louis

On 09/02/11 13:35 +0100, Louis Rilling wrote:
> On 09/02/11 7:18 -0500, Oren Laadan wrote:
> >
> >
> > On 02/09/2011 07:01 AM, Louis Rilling wrote:
> > > On 08/02/11 18:09 -0800, Sukadev Bhattiprolu wrote:
> > >> Oren Laadan [orenl@cs.columbia.edu] wrote:
> > >> |
> > >> |
> > >> | On 02/05/2011 04:40 PM, Sukadev Bhattiprolu wrote:
> > >> | > Oren Laadan [orenl@cs.columbia.edu] wrote:
> > >> | > | Suka,
> > >> | > |
> > >> | > | This patch - and the corresponding kernel patch - are wrong
> > >> | >
> > >> | > Ah, I see that now.
> > >> | >
> > >> | > But am not sure about the kernel part though. We were getting a crash
> > >> | > reliably (with older kernels) because of the ->exit_signal = -1 in
> > >> | > do_ghost_task().
> > >> |
> > >> | Are we still getting it with 2.6.37 ?
> > >>
> > >> I am not currently getting the crash on 2.6.37 - I thought it was due to
> > >> the following commit which removed the check for task_detached() in
> > >> do_wait_thread().
> > >>
> > >> commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
> > >> Author: Oleg Nesterov <oleg@redhat.com>
> > >> Date: Thu Dec 17 15:27:15 2009 -0800
> > >
> > > I don't think that this introduced the bug. The bug triggers with EXIT_DEAD
> > > tasks, for which wait() must ignore (see below). So, the bug looks still there
> > > in 2.6.37.
> > >
> > >>
> > >> But if that is true, I need to investigate why Louis Rilling was getting
> > >> the crash in Jun 2010 - which he tried to fix here:
> > >>
> > >> http://lkml.org/lkml/2010/6/16/295
> > >
> > > I was getting the crash on Kerrighed, which heavily patches the 2.6.30 kernel.
> > > I could reproduce it on vanilla Linux of the moment (2.6.35-rc3), but
> > > only after introducing artificial delays in release_task().
> > >
> > > IIRC, what triggers the crash is some exiting detached task in the

> > > pid_namespace, which goes EXIT_DEAD, and as such cannot be reaped by
> > > zap_pid_ns_processes()->sys_wait4(). So with some odd timing, the detached
> > > task can call proc_flush_task() after container init does, which triggers the
> > > proc_mnt crash.
> > >
> > > Container init Some detached task in the ctnr
> > > exit_notify()
> > > ->exit_state = EXIT_DEAD
> > > exit_notify()
> > > forget_original_parent()
> > > find_new_reaper()
> > > zap_pid_ns_processes()
> > > sys_wait4()
> > > /* cannot reap EXIT_DEAD tasks */
> > > /* reparents EXIT_DEAD tasks to global init */
> > >
> > > Container reaper
> > > release_task()
> > > proc_flush_task()
> > > pid_ns_release_proc()
> > > release_task()
> > > proc_flush_task()
> > > proc_flush_task_mnt()
> > > KABOOM
> >
> > Louis, thanks for the explanation, and two follow-up questions:
> >
> > 1) Is there a patch circulating for this ? or even better, on the
> > way to mainline ?
>
> We finally agreed on a patch from Eric, but for some unknown reason, it has not
> been finalized(?) and routed to mainline yet.
>
> https://lkml.org/lkml/2010/7/12/213
>
> >
> > 2) Would it suffice if the c/r code ensures that the init never
> > exits before any EXIT_DEAD tasks ?
>
> That's what Eric's patch does: make zap_pid_ns_processes() wait until all other
> tasks (EXIT_DEAD or whatever) have passed
> release_task()->__exit_signal()->__unhash_process().
>
> Thanks,
>
> Louis
>
> --

> Dr Louis Rilling Kerlabs
> Skype: louis.rilling Batiment Germanium
> Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
> http://www.kerlabs.com/ 35700 Rennes

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Wed, 09 Feb 2011 19:02:16 GMT
View Forum Message <> Reply to Message

Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
| > | Are we still getting it with 2.6.37 ?
| >
| > I am not currently getting the crash on 2.6.37 - I thought it was due to
| > the following commit which removed the check for task_detached() in
| > do_wait_thread().
| >
| > commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
| > Author: Oleg Nesterov <oleg@redhat.com>
| > Date: Thu Dec 17 15:27:15 2009 -0800
|
| I don't think that this introduced the bug. The bug triggers with EXIT_DEAD
| tasks, for which wait() must ignore (see below). So, the bug looks still there
| in 2.6.37.

Sorry, I did not mean to imply that the above commit caused the crash
you saw in Jun 2010.

I can reproduce a crash with 2.6.32 - where if container-init terminates
before a detached child, we get a crash when the detached child calls
proc_flush_mnt(). I suspected it was because do_wait_thread() skipped
over detached tasks (in 2.6.32).

The same test case does not crash on 2.6.37 - which includes the above commit.
The removes the check for detached tasks, my initial guess is that the above

commit, may have contributed to _fixing_ the crash in 2.6.37.

Sukadev

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Thu, 10 Feb 2011 02:44:30 GMT
View Forum Message <> Reply to Message

Oren Laadan [orenl@cs.columbia.edu] wrote:
|
|
| > But if that is true, I need to investigate why Louis Rilling was getting
| > the crash in Jun 2010 - which he tried to fix here:
| >
| > http://lkml.org/lkml/2010/6/16/295
|
| I see. So basically there is a kerenl bug that can be potentially
| exposed by the c/r code. Therefore, we need to fix the kernel bug...
| (and until such a fix makes it to mainline, we'll add it as part of
| the linux-cr patchset).

Yes, but there is more than one problem (at least in our C/R kernel).

There is the bug that Louis Rilling reported and Eric has a fix for.
Even if we apply Eric's fix to the C/R kernel, we still will have
another problem if do_ghost_task() sets ->exit_signal to -1.

Consider this in 2.6.37:

Container-init: Ghost child of container-init

```
      do_ghost_task()
zap_pid_ns..()
  Send SIGKILL

  do_wait()
    - adds self to ->wait_chldexit queue
    - goes through do_wait_thread() - finds that
      it has at least one child  (on tsk->children),
      but it has not yet exited
    - so waits for the child to exit
        wakes up for SIGKILL
       ->exit_signal = -1
```

do_exit()

Note that exit_notify() does not notify parent when the ghost process
exits, because ->exit_signal is -1.

So you may ask how did the container-init have a ghost child. That was
due to a bug in usercr :-).

But my point is such a userspace bug can leave the above container init
unkillable.

Note that this does not happen with normal threads which set ->exit_signal
to -1 . That is because of the following two pieces of code in copy_process():

```
/* ok, now we should be set up.. */
 p->exit_signal = (clone_flags & CLONE_THREAD) ? -1 : (clone_flags & CSIGNAL);
```

and

```
 /* CLONE_PARENT re-uses the old parent */
 if (clone_flags & (CLONE_PARENT|CLONE_THREAD)) {
        p->real_parent = current->real_parent;
        p->parent_exec_id = current->parent_exec_id;
```

With this our container-init above will not have any children to wait
for in do_wait_thread().

Sukadev

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Thu, 10 Feb 2011 03:53:56 GMT
View Forum Message <> Reply to Message

On 02/09/2011 09:44 PM, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> |
> |
> | > But if that is true, I need to investigate why Louis Rilling was getting
> | > the crash in Jun 2010 - which he tried to fix here:
> | >
> | > http://lkml.org/lkml/2010/6/16/295
> |
> | I see. So basically there is a kerenl bug that can be potentially

> | exposed by the c/r code. Therefore, we need to fix the kernel bug...
> | (and until such a fix makes it to mainline, we'll add it as part of
> | the linux-cr patchset).
>
> Yes, but there is more than one problem (at least in our C/R kernel).
>
> There is the bug that Louis Rilling reported and Eric has a fix for.
> Even if we apply Eric's fix to the C/R kernel, we still will have
> another problem if do_ghost_task() sets ->exit_signal to -1.
>
> Consider this in 2.6.37:
>
> Container-init: Ghost child of container-init
>
>   do_ghost_task()
> zap_pid_ns..()
>   Send SIGKILL
>
>   do_wait()
>   - adds self to ->wait_chldexit queue
>   - goes through do_wait_thread() - finds that
>   it has at least one child (on tsk->children),
>   but it has not yet exited
>   - so waits for the child to exit
>   wakes up for SIGKILL
>   ->exit_signal = -1
>   do_exit()
>
> Note that exit_notify() does not notify parent when the ghost process
> exits, because ->exit_signal is -1.

I see.. nice catch :)

To address this, initially I thought that we could make ghosts take
the tasklist_lock (write) when they change their ->exit_signal.

But that's not enough because the parent may already be blocked in
wait (so it's too late). Therefore, we also need to have ghosts
wake-up their parent through __wake_up_parent().

so something like:

```
void ghost_auto_reapable()
{
 write_lock(&tasklist_lock);
 current->exit_signal = -1;
 __wake_up_sync_key(current, current->parent);
 write_unlock(&tasklist_lock);
```

}

If the parent wasn't at all waiting for us, no harm done...

>
> So you may ask how did the container-init have a ghost child. That was
> due to a bug in usercr :-).

You don't need a bug: the ghost flag is used for both ghost and dead
tasks (the former used to instantiate specific pids, the latter to
make other tasks orphans). So restarting a container that had orphan
tasks is guaranteed to do this.

Oren.

>
> But my point is such a userspace bug can leave the above container init
> unkillable.
>
> Note that this does not happen with normal threads which set ->exit_signal
> to -1 . That is because of the following two pieces of code in copy_process():
>
> /* ok, now we should be set up.. */
> p->exit_signal = (clone_flags & CLONE_THREAD) ? -1 : (clone_flags & CSIGNAL);
>
> and
>
> /* CLONE_PARENT re-uses the old parent */
> if (clone_flags & (CLONE_PARENT|CLONE_THREAD)) {
> p->real_parent = current->real_parent;
> p->parent_exec_id = current->parent_exec_id;
>
> With this our container-init above will not have any children to wait
> for in do_wait_thread().
>
> Sukadev
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Thu, 10 Feb 2011 06:17:30 GMT
View Forum Message <> Reply to Message

Oren Laadan [orenl@cs.columbia.edu] wrote:

|
| To address this, initially I thought that we could make ghosts take
| the tasklist_lock (write) when they change their ->exit_signal.
|
| But that's not enough because the parent may already be blocked in
| wait (so it's too late). Therefore, we also need to have ghosts
| wake-up their parent through __wake_up_parent().
|
| so something like:
|
| void ghost_auto_reapable()
| {
| write_lock(&tasklist_lock);
| current->exit_signal = -1;
| __wake_up_sync_key(current, current->parent);
| write_unlock(&tasklist_lock);

You meant __wake_up_parent() here I guess.

But if we do this in do_ghost_task(), the parent could wakeup, find
that it still has a live child (this ghost) and go back to sleep before
the ghost becomes EXIT_DEAD right ?

If so, we would still have the problem ?

i.e we must stop being a chld a of the cinit for it to not wait for us.
Or we might need to detect that the the pidns is going away, so signalling
the parent won't cause any harm. But that is racy too :-(

| }
|
| If the parent wasn't at all waiting for us, no harm done...
|
| >
| > So you may ask how did the container-init have a ghost child. That was
| > due to a bug in usercr :-).
|
| You don't need a bug: the ghost flag is used for both ghost and dead
| tasks (the former used to instantiate specific pids, the latter to
| make other tasks orphans). So restarting a container that had orphan
| tasks is guaranteed to do this.

Ah, thats a good point.

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Thu, 10 Feb 2011 10:23:12 GMT
View Forum Message <> Reply to Message

On 09/02/11 11:02 -0800, Sukadev Bhattiprolu wrote:
> Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
> | > | Are we still getting it with 2.6.37 ?
> | >
> | > I am not currently getting the crash on 2.6.37 - I thought it was due to
> | > the following commit which removed the check for task_detached() in
> | > do_wait_thread().
> | >
> | > commit 9cd80bbb07fcd6d4d037fad4297496d3b132ac6b
> | > Author: Oleg Nesterov <oleg@redhat.com>
> | > Date: Thu Dec 17 15:27:15 2009 -0800
> |
> | I don't think that this introduced the bug. The bug triggers with EXIT_DEAD
> | tasks, for which wait() must ignore (see below). So, the bug looks still there
> | in 2.6.37.
>
> Sorry, I did not mean to imply that the above commit caused the crash
> you saw in Jun 2010.
>
> I can reproduce a crash with 2.6.32 - where if container-init terminates
> before a detached child, we get a crash when the detached child calls
> proc_flush_mnt(). I suspected it was because do_wait_thread() skipped
> over detached tasks (in 2.6.32).
>
> The same test case does not crash on 2.6.37 - which includes the above commit.
> The removes the check for detached tasks, my initial guess is that the above
> commit, may have contributed to _fixing_ the crash in 2.6.37.

Hm, I don't see how this commit changed things for detached tasks, unless ptrace
is involved. Detached tasks go atomically
from ->exit_state == 0 to ->exit_state == EXIT_DEAD in exit_notify(),
because tracehook_notify_death() returns DEATH_REAP for all not ptraced detached
tasks.

What do you think has changed precisely?

Thanks,

Louis

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Oren Laadan on Thu, 10 Feb 2011 14:56:28 GMT
View Forum Message <> Reply to Message

On 02/10/2011 01:17 AM, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> |
> | To address this, initially I thought that we could make ghosts take
> | the tasklist_lock (write) when they change their ->exit_signal.
> |
> | But that's not enough because the parent may already be blocked in
> | wait (so it's too late). Therefore, we also need to have ghosts
> | wake-up their parent through __wake_up_parent().
> |
> | so something like:
> |
> | void ghost_auto_reapable()
> | {
> | write_lock(&tasklist_lock);
> | current->exit_signal = -1;
> | __wake_up_sync_key(current, current->parent);
> | write_unlock(&tasklist_lock);
>
> You meant __wake_up_parent() here I guess.

Yes...

>
> But if we do this in do_ghost_task(), the parent could wakeup, find
> that it still has a live child (this ghost) and go back to sleep before
> the ghost becomes EXIT_DEAD right ?

You are right again...

>
> If so, we would still have the problem ?
>
> i.e we must stop being a chld a of the cinit for it to not wait for us.
> Or we might need to detect that the the pidns is going away, so signalling
> the parent won't cause any harm. But that is racy too :-(

So instead, we can call __wake_up_parent() from exit_checkpoint()
if indeed we are already reaped there:

```
exit_checkpoint()
{
 ...
 if (current->flags & PF_RESTARTING) {
  ...
  /* either zombie or reaped ghost/dead */
  if (current->exit_state = EXIT_DEAD)
   __wake_up_parent(...);   /* probably need lock */
  ...
 }
 ...
}
```

and to avoid userspace misuse, disallow non-thread-group-leader ghosts.

?

Oren.

> | }
> |
> | If the parent wasn't at all waiting for us, no harm done...
> |
> | >
> | > So you may ask how did the container-init have a ghost child. That was
> | > due to a bug in usercr :-).
> |
> | You don't need a bug: the ghost flag is used for both ghost and dead
> | tasks (the former used to instantiate specific pids, the latter to
> | make other tasks orphans). So restarting a container that had orphan
> | tasks is guaranteed to do this.
>
> Ah, thats a good point.
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Thu, 10 Feb 2011 17:42:18 GMT
View Forum Message <> Reply to Message

Oren Laadan [orenl@cs.columbia.edu] wrote:
|
|
| On 02/10/2011 01:17 AM, Sukadev Bhattiprolu wrote:
| > Oren Laadan [orenl@cs.columbia.edu] wrote:
| > |
| > | To address this, initially I thought that we could make ghosts take
| > | the tasklist_lock (write) when they change their ->exit_signal.
| > |
| > | But that's not enough because the parent may already be blocked in
| > | wait (so it's too late). Therefore, we also need to have ghosts
| > | wake-up their parent through __wake_up_parent().
| > |
| > | so something like:
| > |
| > | void ghost_auto_reapable()
| > | {
| > | write_lock(&tasklist_lock);
| > | current->exit_signal = -1;
| > | __wake_up_sync_key(current, current->parent);
| > | write_unlock(&tasklist_lock);
| >
| > You meant __wake_up_parent() here I guess.

Hmm, can we have the above wakeup and, like in 2.6.32, have do_wait_thread()
skip over detached tasks ? Since we set ->exit_signal above,
do_wait_thread() should not wait for us.

I will go through Oleg's patch earlier again. But my guess, without C/R
do_wait_thread() had no reason to have a detached child in do_wait_thread()
which is probably why Oleg removed it.

I will look at the exit_checkpoint() change you mention later today.

Sukadev

_____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Thu, 10 Feb 2011 17:54:09 GMT
View Forum Message <> Reply to Message

Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
| > I can reproduce a crash with 2.6.32 - where if container-init terminates
| > before a detached child, we get a crash when the detached child calls

| > proc_flush_mnt(). I suspected it was because do_wait_thread() skipped
| > over detached tasks (in 2.6.32).
| >
| > The same test case does not crash on 2.6.37 - which includes the above commit.
| > The removes the check for detached tasks, my initial guess is that the above
| > commit, may have contributed to _fixing_ the crash in 2.6.37.
|
| Hm, I don't see how this commit changed things for detached tasks, unless ptrace
| is involved. Detached tasks go atomically
| from ->exit_state == 0 to ->exit_state == EXIT_DEAD in exit_notify(),
| because tracehook_notify_death() returns DEATH_REAP for all not ptraced detached
| tasks.
|
| What do you think has changed precisely?

Well, one of the changes in the commit is this:

```
@@ -1551,14 +1554,9 @@ static int do_wait_thread(struct wait_opts *wo, struct task_struct *tsk)
        struct task_struct *p;

        list_for_each_entry(p, &tsk->children, sibling) {
-               /*
-                * Do not consider detached threads.
-                */
-               if (!task_detached(p)) {
-                       int ret = wait_consider_task(wo, 0, p);
-                       if (ret)
-                               return ret;
-               }
+               int ret = wait_consider_task(wo, 0, p);
+               if (ret)
+                       return ret;
        }

        return 0;
```

If it was a detached task, do_wait_thread() skipped over it. In the C/R
kernel we were setting the ->exit_signal to -1 for a "ghost" process.
I assumed that the container-init exited without waiting for the "ghost"
and we were getting the crash in proc_flush_mnt() when the ghost exited.

Sukadev

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Thu, 10 Feb 2011 18:04:33 GMT
View Forum Message <> Reply to Message

On 10/02/11 9:54 -0800, Sukadev Bhattiprolu wrote:
> Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
> | > I can reproduce a crash with 2.6.32 - where if container-init terminates
> | > before a detached child, we get a crash when the detached child calls
> | > proc_flush_mnt(). I suspected it was because do_wait_thread() skipped
> | > over detached tasks (in 2.6.32).
> | >
> | > The same test case does not crash on 2.6.37 - which includes the above commit.
> | > The removes the check for detached tasks, my initial guess is that the above
> | > commit, may have contributed to _fixing_ the crash in 2.6.37.
> |
> | Hm, I don't see how this commit changed things for detached tasks, unless ptrace
> | is involved. Detached tasks go atomically
> | from ->exit_state == 0 to ->exit_state == EXIT_DEAD in exit_notify(),
> | because tracehook_notify_death() returns DEATH_REAP for all not ptraced detached
> | tasks.
> |
> | What do you think has changed precisely?
>
> Well, one of the changes in the commit is this:
>
> @@ -1551,14 +1554,9 @@ static int do_wait_thread(struct wait_opts *wo, struct task_struct
*tsk)
> 	struct task_struct *p;
>
> 	list_for_each_entry(p, &tsk->children, sibling) {
> -		/*
> -		 * Do not consider detached threads.
> -		 */
> -		if (!task_detached(p)) {
> -			int ret = wait_consider_task(wo, 0, p);
> -			if (ret)
> -				return ret;
> -		}
> +		int ret = wait_consider_task(wo, 0, p);
> +		if (ret)
> +			return ret;
> 	}
>
> 	return 0;
>
> ---
> If it was a detached task, do_wait_thread() skipped over it. In the C/R
> kernel we were setting the ->exit_signal to -1 for a "ghost" process.
> I assumed that the container-init exited without waiting for the "ghost"

> and we were getting the crash in proc_flush_mnt() when the ghost exited.

The point is that wait_consider_task() skips detached tasks as soon as they are
not ptraced. So removing the check in do_wait_thread() should not have changed
the behavior. Am I missing something?

Thanks,

Louis

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Thu, 10 Feb 2011 22:31:59 GMT
View Forum Message <> Reply to Message

Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
| On 10/02/11 9:54 -0800, Sukadev Bhattiprolu wrote:
| > Louis Rilling [Louis.Rilling@kerlabs.com] wrote:

| > If it was a detached task, do_wait_thread() skipped over it. In the C/R
| > kernel we were setting the ->exit_signal to -1 for a "ghost" process.
| > I assumed that the container-init exited without waiting for the "ghost"
| > and we were getting the crash in proc_flush_mnt() when the ghost exited.
|
| The point is that wait_consider_task() skips detached tasks as soon as they are
| not ptraced. So removing the check in do_wait_thread() should not have changed
| the behavior. Am I missing something?

No. I was :-) You are right that it did not change the behavior. I still
need to investigate why the crash does not occur on 2.6.37 even without
Eric's fix.

Sukadev

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Wed, 16 Feb 2011 20:10:20 GMT
View Forum Message <> Reply to Message

Oren Laadan [orenl@cs.columbia.edu] wrote:
| So instead, we can call __wake_up_parent() from exit_checkpoint()
| if indeed we are already reaped there:
|
| exit_checkpoint()
| {
| ...
| if (current->flags & PF_RESTARTING) {
| ...
| /* either zombie or reaped ghost/dead */
| if (current->exit_state = EXIT_DEAD)
| __wake_up_parent(...);   /* probably need lock */
| ...
| }
| ...
| }
|
| and to avoid userspace misuse, disallow non-thread-group-leader ghosts.
|
| ?

Well, I don't see a problem as such, but notice one inconsistency.

By the time the ghost task calls exit_checkpoint() it would have
gone through release_task()/__exit_signal()/__unhash_process() so
it is no longer on the parent's ->children list. We will be accessing
the task's ->parent pointer after this.

I am looking to see if anything prevents the parent from itself going
through release_task(), after the child does the release_task() but before
the child does the exit_checkpoint().

In 2.6.38, I don't see specifically where a task's ->parent pointer is
invalidated. The task->parent and task->parent->signal are freed in the
final __put_task_struct(). So its probably safe to access them, even if the
parent itself is exiting and has gone through release_task().

But in 2.6.32 i.e RHEL5, tsk->signal is set to NULL in __exit_signal().
So, I am trying to rule out the following scenario:

 Child (may not be a ghost)   Parent
 -------------------------   ------
- exit_notify(): is EXIT_DEAD
- release_task():
  - drops task_list_lock

- itself proceeds to exit.
- enters release_task()
- sets own->signal = NULL
(in 2.6.32, __exit_signal())

- enters exit_checkpoint()
- __wake_up_parent()
access parents->signal NULL ptr

Not sure if holding task_list_lock here is needed or will help.

Sukadev

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Thu, 17 Feb 2011 15:21:16 GMT
View Forum Message <> Reply to Message

On 16/02/11 12:10 -0800, Sukadev Bhattiprolu wrote:
> Oren Laadan [orenl@cs.columbia.edu] wrote:
> | So instead, we can call __wake_up_parent() from exit_checkpoint()
> | if indeed we are already reaped there:
> |
> | exit_checkpoint()
> | {
> | ...
> | if (current->flags & PF_RESTARTING) {
> | ...
> | /* either zombie or reaped ghost/dead */
> | if (current->exit_state = EXIT_DEAD)
> | __wake_up_parent(...); /* probably need lock */
> | ...
> | }
> | ...
> | }
> |
> | and to avoid userspace misuse, disallow non-thread-group-leader ghosts.
> |
> | ?
>
> Well, I don't see a problem as such, but notice one inconsistency.
>
> By the time the ghost task calls exit_checkpoint() it would have
> gone through release_task()/__exit_signal()/__unhash_process() so

> it is no longer on the parent's ->children list. We will be accessing
> the task's ->parent pointer after this.
>
> I am looking to see if anything prevents the parent from itself going
> through release_task(), after the child does the release_task() but before
> the child does the exit_checkpoint().
>
> In 2.6.38, I don't see specifically where a task's ->parent pointer is
> invalidated. The task->parent and task->parent->signal are freed in the
> final __put_task_struct(). So its probably safe to access them, even if the
> parent itself is exiting and has gone through release_task().
>
> But in 2.6.32 i.e RHEL5, tsk->signal is set to NULL in __exit_signal().
> So, I am trying to rule out the following scenario:
>
> Child (may not be a ghost)   Parent
> ------------------------   ------
> - exit_notify(): is EXIT_DEAD
> - release_task():
>   - drops task_list_lock
>   - itself proceeds to exit.
>   - enters release_task()
>   - sets own->signal = NULL
>   (in 2.6.32, __exit_signal())
>
> - enters exit_checkpoint()
> - __wake_up_parent()
> access parents->signal NULL ptr
>
> Not sure if holding task_list_lock here is needed or will help.

Giving my 2 cents since I've been Cc'ed.

AFAICS, holding tasklist_lock prevents __exit_signal() from setting
parent->signal to NULL in your back. So something like this should be safe:

```
 read_lock(&tasklist_lock);
 if (current->parent->signal)
   __wake_up_parent(...);
 read_unlock(&tasklist_lock);
```

I haven't looked at the context, but of course this also requires that some
get_task_struct() on current->parent has been done somewhere else before current
has passed __exit_signal().

By the way, instead of checking current->parent->signal,
current->parent->exit_state would look cleaner to me. current->parent is not
supposed to wait on ->wait_chldexit after calling do_exit(), right?

Louis

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Sukadev Bhattiprolu on Mon, 21 Feb 2011 20:40:58 GMT
View Forum Message <> Reply to Message

Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
| > But in 2.6.32 i.e RHEL5, tsk->signal is set to NULL in __exit_signal().
| > So, I am trying to rule out the following scenario:
| >
| > Child (may not be a ghost) Parent
| > ------------------------ ------
| > - exit_notify(): is EXIT_DEAD
| > - release_task():
| > - drops task_list_lock
| > - itself proceeds to exit.
| > - enters release_task()
| > - sets own->signal = NULL
| > (in 2.6.32, __exit_signal())
| >
| > - enters exit_checkpoint()
| > - __wake_up_parent()
| > access parents->signal NULL ptr
| >
| > Not sure if holding task_list_lock here is needed or will help.
|
| Giving my 2 cents since I've been Cc'ed.

Thanks, appreciate the input :-)

|
| AFAICS, holding tasklist_lock prevents __exit_signal() from setting
| parent->signal to NULL in your back. So something like this should be safe:
|
| read_lock(&tasklist_lock);

| if (current->parent->signal)
| __wake_up_parent(...);
| read_unlock(&tasklist_lock);

Yes, checking the parent->signal with task_list_lock would work.

|
| I haven't looked at the context, but of course this also requires that some
| get_task_struct() on current->parent has been done somewhere else before current
| has passed __exit_signal().
|
| By the way, instead of checking current->parent->signal,
| current->parent->exit_state would look cleaner to me. current->parent is not
| supposed to wait on ->wait_chldexit after calling do_exit(), right?
^^^^^^^

Hmm, do you mean exit_notify() here ?

If so, yes checking the exit_state is cleaner.

If the parent's exit_state is set, then it can't be waiting for the ghost,
so no need to wake_up_parent(). If exit state is not set, then it is safe
to wake_up_parent() (parent->signal would not yet have been cleared for
instance).

The one case where a parent in do_exit() could still wait for the child is
the container-init which waits on wait_chldexit in do_exit() ->
zap_pid_ns_processes() - but even in that case the __wake_up_parent()
call would be safe.

Sukadev

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs

---

Subject: Re: [PATCH][usercr]: Ghost tasks must be detached
Posted by Louis Rilling on Tue, 22 Feb 2011 10:28:41 GMT
View Forum Message <> Reply to Message

On 21/02/11 12:40 -0800, Sukadev Bhattiprolu wrote:
> Louis Rilling [Louis.Rilling@kerlabs.com] wrote:
> | > But in 2.6.32 i.e RHEL5, tsk->signal is set to NULL in __exit_signal().
> | > So, I am trying to rule out the following scenario:
> | >
> | > Child (may not be a ghost) Parent
> | > ------------------------ ------

> | > - exit_notify(): is EXIT_DEAD
> | > - release_task():
> | >   - drops task_list_lock
> | >   - itself proceeds to exit.
> | >   - enters release_task()
> | >   - sets own->signal = NULL
> | >   (in 2.6.32, __exit_signal())
> | >
> | > - enters exit_checkpoint()
> | > - __wake_up_parent()
> | > access parents->signal NULL ptr
> | >
> | > Not sure if holding task_list_lock here is needed or will help.
> |
> | Giving my 2 cents since I've been Cc'ed.
>
> Thanks, appreciate the input :-)
>
> |
> | AFAICS, holding tasklist_lock prevents __exit_signal() from setting
> | parent->signal to NULL in your back. So something like this should be safe:
> |
> | read_lock(&tasklist_lock);
> | if (current->parent->signal)
> | __wake_up_parent(...);
> | read_unlock(&tasklist_lock);
>
> Yes, checking the parent->signal with task_list_lock would work.
>
> |
> | I haven't looked at the context, but of course this also requires that some
> | get_task_struct() on current->parent has been done somewhere else before current
> | has passed __exit_signal().
> |
> | By the way, instead of checking current->parent->signal,
> | current->parent->exit_state would look cleaner to me. current->parent is not
> | supposed to wait on ->wait_chldexit after calling do_exit(), right?
>   ^^^^^^^
>
> Hmm, do you mean exit_notify() here ?

Right, I had forgotten zap_pid_ns_processes() ;) My point was just that once
->exit_state is set (for all threads), ->signal->wait_chldexit is not used
anymore. But I'm sure that you got it right :)

Thanks,

Louis

>
> If so, yes checking the exit_state is cleaner.
>
> If the parent's exit_state is set, then it can't be waiting for the ghost,
> so no need to wake_up_parent(). If exit state is not set, then it is safe
> to wake_up_parent() (parent->signal would not yet have been cleared for
> instance).
>
> The one case where a parent in do_exit() could still wait for the child is
> the container-init which waits on wait_chldexit in do_exit() ->
> zap_pid_ns_processes() - but even in that case the __wake_up_parent()
> call would be safe.
>
> Sukadev
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containe rs

--
Dr Louis Rilling Kerlabs
Skype: louis.rilling Batiment Germanium
Phone: (+33|0) 6 80 89 08 23 80 avenue des Buttes de Coesmes
http://www.kerlabs.com/ 35700 Rennes

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs